A
Project Report
On
"Crop Leaf Disease Detection"

Prepared by

Maan Patel (21DCS078)
Deep Rohit (21DCS105)

Under the guidance of

Dr. Nilesh Dubey

Associate Professor

Krishna Patel

Assistant Professor

A Report Submitted to

Charotar University of Science and Technology

for Partial Fulfillment of the Requirements for the

5th Semester Software Group Project-III (CS348)

Submitted at

Computer Science and Engineering

Devang Patel Institute of Advance Technology and Research (DEPSTAR)

Faculty of Technology & Engineering (FTE), CHARUSAT

At: Changa, Dist: Anand – 388421

October 2023

DECLARATION BY THE CANDIDATE

We hereby declare that the project report entitled “Crop Leaf Disease Detection” submitted by us to the Devang Patel Institute of Advance Technology and Research, Changa in partial fulfilment of the requirement for the award of the degree of B.Tech in Computer Science & Engineering, from DEPSTAR/FTE, is a record of a bonafide Software Group Project-III carried out by us under the guidance of Associate Prof. Dr. Nilesh Dubey and Assistant Prof. Krishna Patel. We further declare that the work carried out and documented in this project report had not been submitted anywhere else, either in part or in full, and it is the original work for the award of any other degree or diploma in this institute or any other institute or university.

Maan Patel (21DCS078)

Deep Rohit(21DCS105)

Dr. Nilesh Dubey	Krishna Patel
Associate Professor	Assistant Professor
Department of Computer Science & Engineering,	Department of Computer Science & Engineering,
DEPSTAR/FTE ,CHARUSAT – CHANGA.	DEPSTAR/FTE ,CHARUSAT – CHANGA.

ABSTRACT

In India, agriculture constitutes a paramount portion of the GDP, underscoring its critical role in the nation's economy. However, the pervasive issue of crop diseases exerts a profound impact on agricultural productivity, resulting in substantial economic losses and jeopardizing global food security. The deleterious effects of these diseases encompass not only reduced yields, but also compromised quality and quantity of agricultural products. Swift and accurate diagnosis of plant diseases through advanced, automated detection techniques holds the key to enhancing food production quality and mitigating economic losses. Leveraging a comprehensive dataset comprising 87,867 images, encompassing 38 categories of healthy and diseased plant leaves gathered under controlled conditions, a sophisticated deep convolutional neural network (CNN) model was trained for the precise identification and classification of various plant species. This system demonstrates impressive capabilities, discerning diseases in 14 different crops including apple, blueberry, acerola, corn, among others. Evaluation of the model's performance, measured in terms of classification accuracy and mean score, revealed an outstanding overall accuracy of approximately 97%, a testament to the effectiveness of the CNN-based approach. This integrated system not only addresses critical agricultural challenges but also represents a significant stride towards ensuring sustainable food security.

ACKNOWLEDGEMENT

We, the developer’s of “Crop Leaf Disease Detection”, with immense pleasure and commitment would like to present the project . The development of this project has given me wide opportunity to think, implement and interact with various aspects of management skills as well as the new emerging technologies.

Every work that one completes successfully stands on the constant encouragement, good will and support of the people around. I hereby avail this opportunity to express my gratitude to number of people who extended their valuable time, full support and cooperation in developing the project.

I express deep sense of gratitude towards our Head of the CSE Department, Associate Professor. Dr. Chirag Patel and project guides Associate Professor. Dr. Nilesh Dubey and Assistant Professor. Krishna Patel for the support during the whole session of study and development. It is because of them, that I was prompted to do hard work, adopting new technologies.

I am sincerely thankful to all the people who helped me complete the project in one way or the other.

Thanks,

Maan Patel (21DCS078)

Deep Rohit (21DCS105)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

INTRODUCTION

The identification of plant diseases is essential for preventing yield and quantity losses in agricultural products. Plant disease research refers to the examination of visually discernible patterns in plants. Plant health monitoring and disease detection are crucial for long-term agriculture. It is extremely difficult to manually monitor plant diseases. It necessitates significant work, expertise in plant diseases, and an extended processing time. As a result, image processing and machine learning techniques are employed in the detection of plant diseases. Image acquisition, image pre-processing, picture segmentation, feature extraction, and classification are all processes in the disease detection process.

1.1 MACHINE LEARNING

The study of mathematical models and methods used by computer systems to gradually increase performance on a given task is known as machine learning (ML). To generate forecasts or choices without being explicitly taught to do so, machine learning algorithms create a mathematical model from sample data, also referred to as "training data".

Machine learning, an area that involves using computers to make predictions, has strong connections to computational statistics. Mathematical optimization plays a significant role in the field of machine learning by providing methods, theories, and applications. Additionally, data mining, a subfield of machine learning, emphasizes exploratory data analysis through unsupervised learning.

In the realm of machine learning, various categories exist to classify different tasks. One prominent category is supervised learning, wherein an algorithm constructs a mathematical model based on a dataset that comprises both input and output data. To illustrate this, imagine a task involving the identification of a specific object within an image. The training data for a supervised learning algorithm would consist of images containing the object in question (input) and corresponding labels designating whether the object is present (output). It's worth noting that in certain scenarios, the input data may be partially available or limited to specialized feedback. Another category, semi-supervised learning, involves creating mathematical models using incomplete training data, where some input samples lack the desired output information.

Plant disease detection via visual means is a more time-consuming and inaccurate task that is only possible in a few places. As opposed to this, using an automatic detection method will need less work, less time, and be more accurate. Brown and yellow spots, early and late scorch, as well as fungal, viral, and bacterial infections are examples of common diseases that affect plants. Image processing is utilized to quantify the disease-affected region and identify the variation in the hue of the afflicted area.

Image segmentation is the process of separating or grouping an image into different parts. There are currently many different ways of performing image segmentation, ranging from the simple thresholding method to advanced color image segmentation methods. These parts normally correspond to something that humans can easily separate and view as individual objects. Computers have no means of intelligently recognizing objects, so many different methods have been developed to segment images. The segmentation process is based on various features found in the image. This might be color information, boundaries, or segment of an image. The image is segmented using the K-means clustering technique. Then unnecessary part (green area) within the leaf area is removed. After that, we calculate the texture features for the segmented infected object. Finally, the extracted features are passed through a pre-trained neural network.

1.2 IMAGE PROCESSING

Image processing is any form of processing for which the input is an image or a series of images or videos, such as photographs or frames of video. The output of image processing can be either an image or a set of characteristics or parameters related to the image. It also means "Analysing and manipulating images with a computer". Image processing is performed this three steps:

- First, import images with an optical device like a scanner or a camera or directly through digital processing.
- Second, manipulate or analyze the images in some way. This step can include image improvement and data summary, or the images are analyzed to find rules that aren't seen by the human eyes. For example, meteorologists use this processing to analyze satellite photographs.
- Last, output the result of image processing. The result might be the image changed in some way or it might be a report based on analysis or result of the images.

1.3 Disease classification with Convolutional neural network(CNN)

Artificial neural networks do incredibly well in machine learning. Artificial neural networks are used to classify words, audio, and images among other things. Different types of neural networks are employed for various tasks; for example, we use recurrent neural networks—more specifically, an LSTM—to predict the order of words, while we use convolutional neural networks to classify images. We will create the fundamental building blocks for CNN in this blog.

ConvNet's name comes from the "convolution" operator. The main purpose of convolution in the case of ConvNet is to extract features from the input image. Convolution preserves the spatial relationships between pixels by using small filters to learn image features.

There are many important parts to a convolutional network. These include the following properties:

1. **Depth:** Depth corresponds to the number of filters we use for the convolution operation.

2. **Stride:** Stride is the number of pixels by which we slide our filter matrix over the input matrix.

3. **Zero-padding:** Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.

4. **Non-Linearity:** ReLU is an element-wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation element-wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear-function like ReLU).

5. **Spatial Pooling:** Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum, etc. These networks have grown in the number of layers leading to architectures such as RestNet and AlexNet that have been trained on images such as Cifar-10 and then tuned to other problems, such as plant classification.

1.4 EXISTING SYSTEM

Describing leaf shape is the biggest drawback in leaf identification. We have extracted several shape options so far to describe the shape of the leaves. However, there is no correct application to classify leaves after the leaf image is captured and its attributes are identified. Plant leaf classification involves classifying leaves based on completely different morphological characteristics.

Many of the classification techniques used are:

- Fuzzy logic
- Principal component Analysis
- k-Nearest Neighbours Classifier

1.5 PROPOSED SYSTEM

The main objective of the proposed system is to detect leaf diseases in plants using feature extraction techniques that consider features such as shape, color, and texture. Convolutional Neural Networks (CNN), a machine learning technique, is used to classify plant leaves into healthy and diseased leaves. If a plant's leaves are diseased, CNN will tell you the name of that specific disease. Treatments for specific diseases are suggested to help grow healthy plants and increase productivity.

To achieve better results and efficiency, different leaf images are first taken with a high-resolution camera. Image processing techniques are then applied to these images to extract useful features for further analysis. The basic steps of the system are summarized as follows:

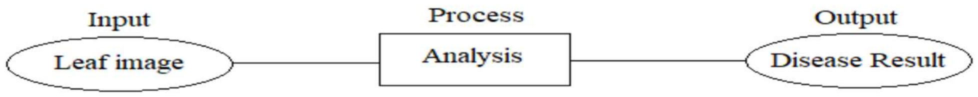


Figure 1: Proposed System.

The advantages of the proposed algorithm are:

- Use an estimator to automatically initialize cluster centers so that no user input is required during segmentation.
- The proposed algorithm improves detection accuracy.
- The proposed method is fully automatic, whereas existing methods require user input to select the optimal segmentation of the input image.
- It also provides environmentally friendly measures to treat the identified diseases.

1.6 OBJECTIVES

- Improve the given input image through image capture and image preprocessing.
- Identify affected areas through texture analysis and segmentation.
- Classify healthy and affected leaf parts through feature extraction and classification.
- Train the model using test data to get accurate results.

1.7 PURPOSE

India is an agricultural country and most of the population depends on agricultural products. Cultivation can be improved with technical support. Diseases can be caused by plant pathogens under any environmental conditions. Disease detection plays an important role in the successful cultivation of crops since in most cases diseases can be detected on the leaves of plants.

There are many techniques for detecting various types of plant diseases at an early stage. Traditional plant disease detection methods rely on visual observation and are ineffective on large crops. Using digital image processing and machine learning, plant disease detection can be done efficiently, quickly, and accurately. This technology saves time, effort, labor, and pesticide use. We hope that this initiative can contribute even a little to agriculture.

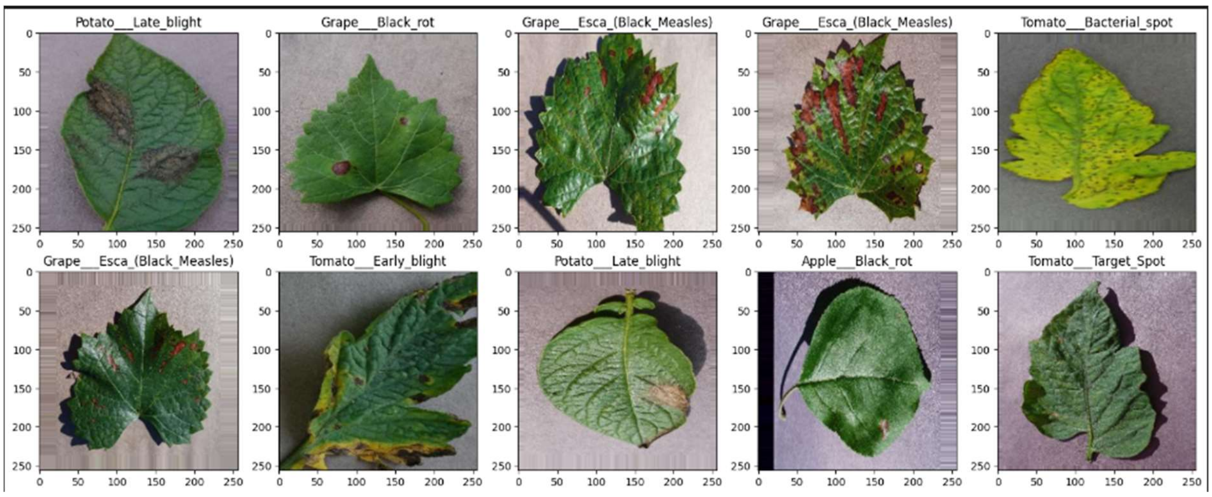


Figure 2: Leaf Disease Classification

CHAPTER 2: LITERATURE SURVEY

LITERATURE SURVEY

2.1 Bharath, S., Kumar, K.V., Pavithran, R., Malathi, T.

The paper "Diagnosing leaf diseases using convolutional neural networks" presents an automatic disease detection system on plant leaves, providing farmers around the world with quick and accurate disease diagnosis. The proposed system focuses on disease identification in major crops such as maize, sugar, wheat, and grapes using the Mo-bileNet model, a type of convolutional neural network (CNN). The study highlights the importance of early detection of agricultural diseases to minimize damage caused by pests or nutrient deficiencies. Traditional manual disease detection methods are time-consuming and less accurate. Therefore, modern techniques using image processing and deep learning technologies are needed. This article reviews the relevant literature on CNN's crop disease detection, highlighting methods and results from previous stud-ies. The authors then present their recommendation system, which uses the TensorFlow and Keras libraries to preprocess the data and train the CNN. Experimental results show high accuracy in disease classification for some crops, while others require further im-provement in training and dataset size. Overall, the proposed system is a promising method to help farmers effectively manage crop health.

2.2 Zeng, Weihui, and Miao Li

This research introduces a novel approach, the "Self-Attention Convolutional Neural Network (SACNN)", for efficient and reliable leaf disease recognition. The difficulties posed by poor contrast between scores and medical history and complex histories that limit recognition accuracy are discussed. SACNN is made up of two networks: one that extracts generic features and one that uses self-attention to zero in on details specific to lesions. Extensive testing on the AES-CD9214 and MK-D2 datasets shows that the recognition accuracy of SACNN is 95.33% and 98.0%, respectively, better than the modern 2.9 methods. %. SAC-NN's self-attention mechanism allows it to identify im-portant image regions, thereby improving recognition accuracy and reliability. The authors discuss the influence of self-attention network parameters on recognition perfor-mance, providing insights for future research. SACNN exhibits strong anti-interference ability when tested with noisy AES-CD9214 images. Overall, the proposed SACNN offers high accuracy and reliability in the identification of crop diseases, making it val-uable for agricultural applications.

2.3 Paymode, A.S., Malode, V.B.

This paper highlights the use of AI in agriculture to detect and classify foliar diseases, especially in tomatoes and grapes. The proposed method using CNN, including the VGG model, achieved high accuracy (98.40% for grapes and 95.71% for tomatoes) in classifying diseased leaves. Research highlights the importance of early disease detection and classification to support agricultural advancement and increase food production.

2.4 Pantazi, X.E., Moshou, D., Tamouridou, A.A

This study introduces a mechanized methodology for the identification of plant diseases through the analysis of picture features and the implementation of single-class classifi-cation. The approach employed in this study involves the utilization of local binary (LBP) models to extract features. Furthermore, it employs dedicated single-class clas-sifiers for each specific plant health condition, including healthy, late blight, and dis-eased states. Algorithms trained on grape leaves exhibit highly generalizable behavior, successfully identifying diseases in different crops. The proposed method achieved an impressive overall success rate of 95% for the 46 combinations of plant conditions tested. Conflict resolution techniques are also introduced to deal with ambiguous data patterns. The study highlights the potential of AI-based methods to accurately and ef-ficiently diagnose plant diseases in agriculture.

2.5 Jaware, T.H., Badgujar, R.D., Patil, P.G

This paper presents an image segmentation technique to detect tree diseases using the K-Means clustering algorithm. The proposed method effectively identifies and con-ceals predominantly green pixels in the leaf image, focusing on specific disease areas. The algorithm removes noisy and boundary data from infected clusters, allowing for more accurate disease classification and identification. The experimental results demonstrate the robustness and effectiveness of the proposed technique, making it a promising tool for detecting foliar diseases in agriculture.

2.6 Sardogan, M., Tuncer, A., Ozen, Y.

This paper presents a method to detect and classify tomato leaf diseases using the CNN model and LVQ algorithm. The dataset contains 500 images with four disease symp-toms. The CNN model extracts features from the RGB and LVQ components that act as classifiers. The results show that it is effective in identifying 4 diseases on tomato leaves. This approach has the potential to detect agricultural diseases early, thereby improving crop yields and management.

2.7 Memon, Muhammad Suleman, Pardeep Kumar, and Rizwan Iqbal

This study presents a novel deep learning metamodel that aims to effectively and pre-cisely detect illnesses occurring on cotton leaves. The dataset comprises a total of 2385 photos depicting both healthy and diseased leaves, which have been augmented to enhance overall performance. The proposed model integrates meta CNN, VGG16 Trans-fer Learning, ResNet50, and a proprietary Deep Learn approach to attain a classification accuracy of 98.53% on the Cotton dataset. Deep learning and transfer learning are cru-cial components in the realm of plant health assessment and disease identification. The primary objective of this study is to establish a comprehensive and dependable frame-work for the accurate detection and classification of several illnesses affecting cotton leaves.

2.8 Liu, Jun, and Xuewei Wang

This review discusses the use of deep learning for plant pest detection, compares it with traditional methods, and highlights its advantages over functional machine learn-ing. It describes plant disease research and deep learning-based pest detection, includ-ing classification, detection, and segmentation networks. Generic datasets and perfor-mance comparisons are provided. Practical application challenges are discussed and potential solutions and future trends are proposed.

2.9 Mohanty, S.P., Hughes, D.P., Salathé, M

This study investigates the application of deep learning technology in the identification of agricultural diseases through picture analysis, which poses a significant challenge to the global food security. A deep convolutional neural network was employed to train on a dataset consisting of 54,306 photos, with the objective of accurately identifying 14 distinct crop species and 26 different illnesses. The most optimal model demonstrated a remarkable accuracy of 99.34% in a sequence of successful evaluations. This study showcases the capacity of utilizing smartphones for disease diagnostics on a global scale, hence providing advantages to small-scale farmers and enhancing global food production.

2.10 Kulkarni, P., Karwande, A., Kolhe, T., Kamble, S., Joshi, A., Wyawahare, M

This study proposes an efficient plant disease detection technique that uses image pro-cessing and machine learning technology with 93° accuracy to detect 20 diseases on 5 common crops. This method consists of pre-processing the RGB image through a Gaussian filter, Otsu threshold, and morphological transformations to extract the fore-ground. The shape, texture, and color features are then extracted and the HSV color space is used to quantify the green color of the image. The publicly available Plant Village dataset is used for testing purposes, providing an efficient and cost-effective computational solution for large-scale crop disease monitoring.

CHAPTER 3: SYSTEM REQUIREMENTS SPECIFICATION

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Hardware Requirements

Processor:	2.5 gigahertz (GHz) frequency or above.
RAM :	A minimum of 4 GB of RAM.
Hard disk:	A minimum of 20 GB of available space.
Input Device:	High resolution camera
Monitor:	Minimum Resolution 1024 X 768.

3.2 Software Requirements

- Operating System: Windows 10 and above.
- Programing language: Python 3.
- Platform: Jupyter Notebook Visual Studio code.
- Supporting libraries: Tensorflow, Keras, SKlearn etc.

3.3 Functional requirements

Functional requirements capture the intended behaviour of the system. This behaviour may be expressed as services, tasks, or the functions the system is required to perform. This lays out important concepts and discusses capturing functional requirements in such a way they can drive architectural decisions and be used to validate the architecture. Features may be additional functionality, or differ from basic functionality along some quality attribute. In the proposed system, concert assesses the compliance of a workflow by analysing the five established elements required to check for the rule adherence in workflows: activities, data, location, resources, and time limits. A rule describes which activities may, must or must not be performed on what objects by which roles. In addition, a rule can further prescribe the order of activities i.e. which activities have to happen before or after other activities.

3.4 Non-Functional requirements

1. Security

- System needs to control the user access and session
- It needs to store the data in a secure location and stored in a secure format
- It requires a secure communication channel for the data.

2. Concurrency and Capacity

System should be able to handle multiple computations executing simultaneously, and potentially interacting with each other.

3. Performance

Performance is generally perceived as a time expectation. This is one of the most important considerations especially when the project is in the architecture phase.

4. Reliability

It is necessary to ensure and notify about the system transactions and processing as simple as keep a system log will increase the time and effort to get it done from the very beginning. Data should be transferred in a reliable way and using trustful protocols.

5. Maintainability

Well-done system is meant to be up and running for long time. Therefore, it will regularly need preventive and corrective maintenance. Maintenance might signify scalability to grow and improve the system features and functionalities.

6. Usability

End user satisfaction and acceptance is one of the key pillars that support a project success. Considering the user experience requirements from the project conception is a win bet, and it will especially save a lot of time at the project release, as the user will not ask for changes or even worst misunderstandings.

7. Documentation

All projects require a minimum of documentation at different levels. In many cases the users might even need training on it, so keeping good documentation practices and standards will do this task spread along the project development; but as well this must be establish since the project planning to include this task in the list.

CHAPTER 4: SYSTEM DESIGN

4.1 Data flow Diagram

A data flow Diagram is a graphical representation of the “flow” of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.

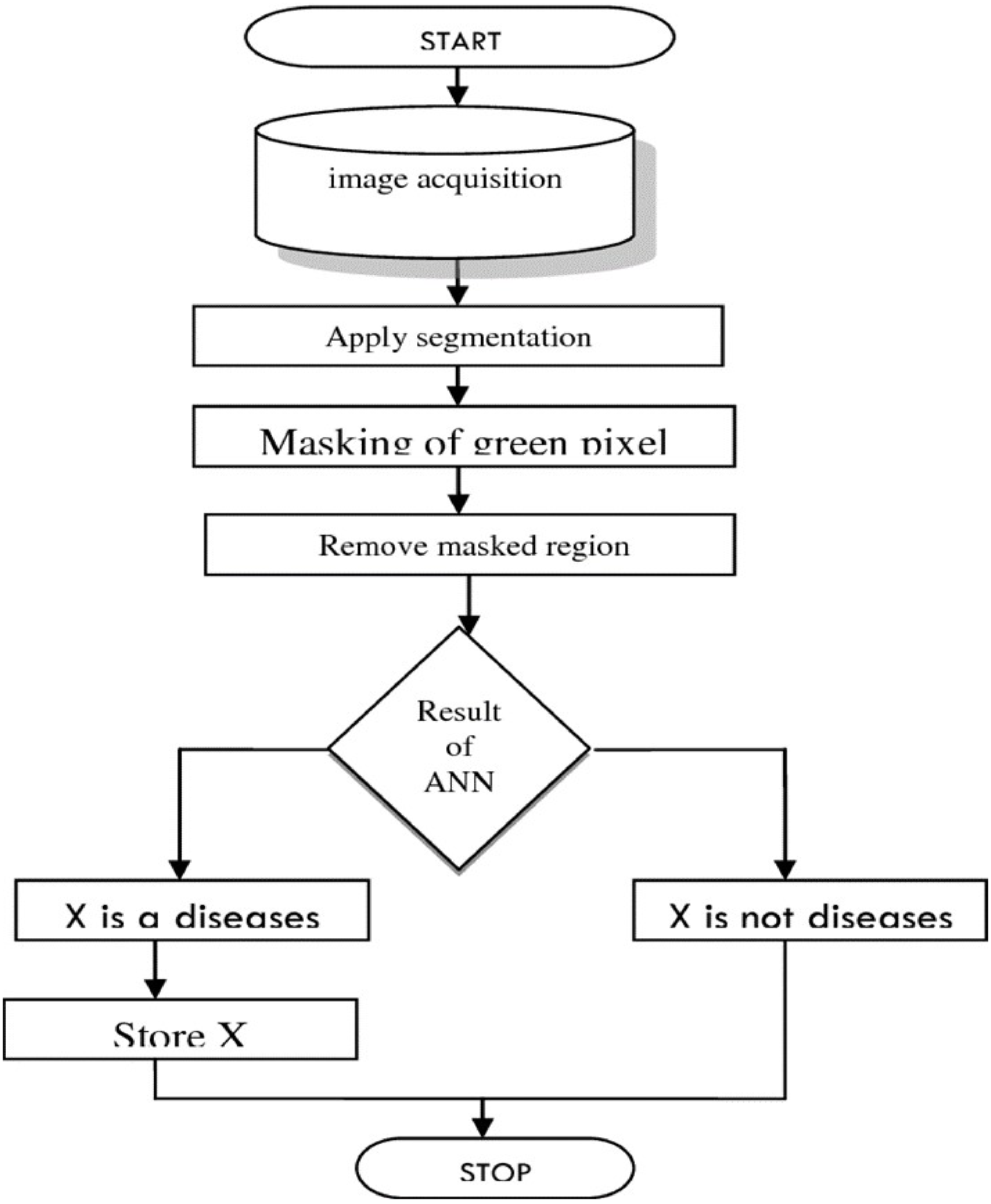


Figure 3: Flowchart for Leaf Classification

4.2 Use case Diagram

Use case diagram is a graphic depiction of the interactions among the elements of a system. Use cases will specify the expected behaviour, and the exact method of making it happen. Use cases once specified can be denoted both textual and visual representation.

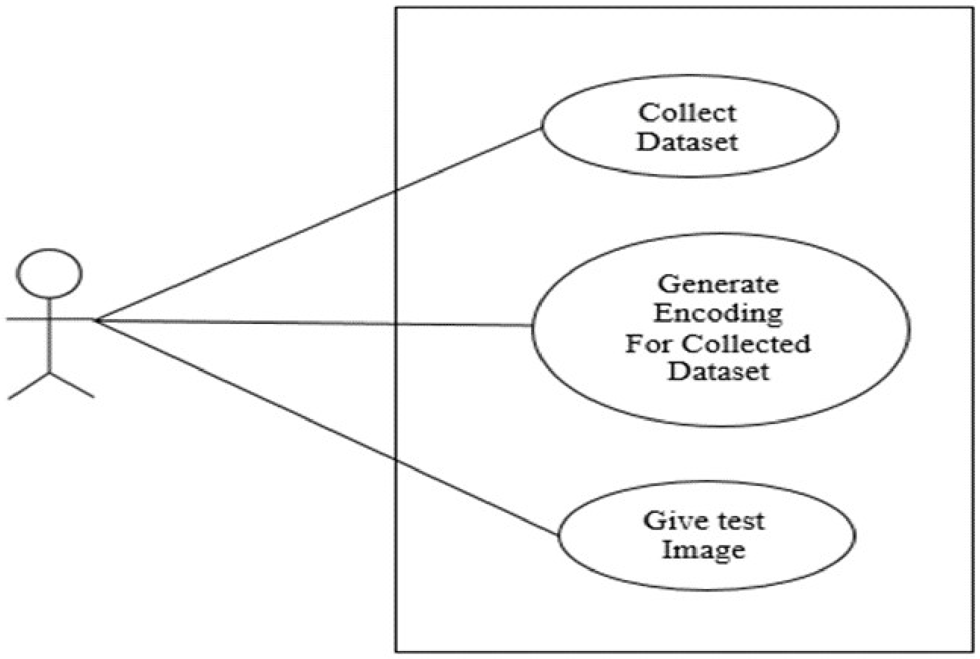


Figure 4: Use Case Diagram

Use case diagrams are used to specify:

- Requirements (external), required usages of a system under design or analysis - to capture what the system is supposed to do.
- The functionality offered by a subject – what the system can do.
- Requirements the specified subject poses on its environment - by defining how environment should interact with the subject so that it will be able to perform its services.

4.3 Sequence Diagram

Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows as parallel vertical lines (lifelines), different processes or objects that live simultaneously and as horizontal arrows, the messages exchanged between them in the order in which they occur.

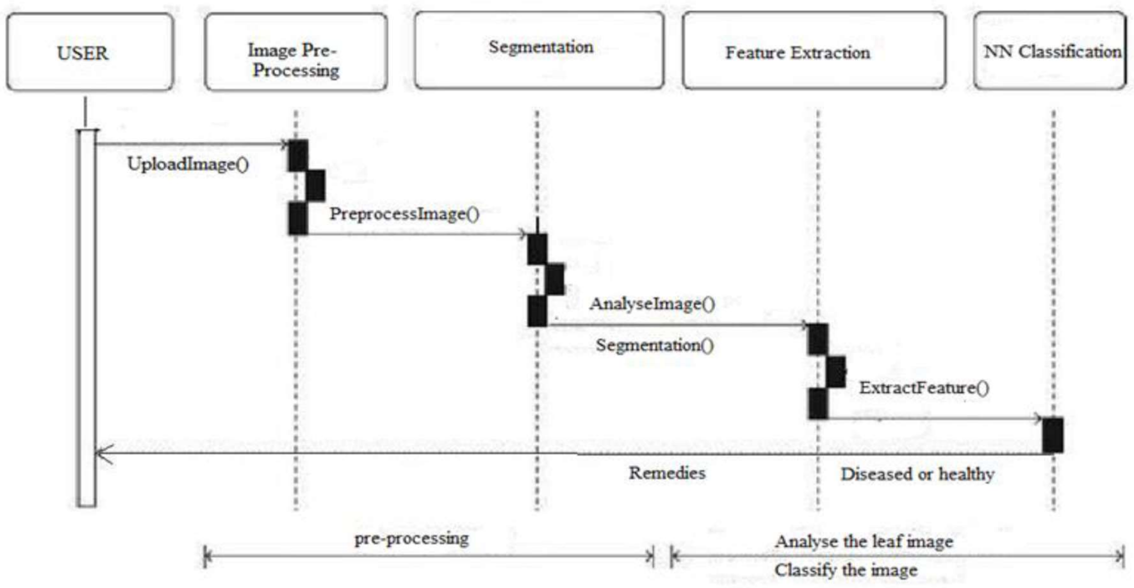


Figure 5: Sequence Diagram

1. **Usage scenarios**: A usage scenario is a description of a potential way your system is used. The logic of a usage scenario may be part of a use case, perhaps an alternate course. It may also be one entire pass through a use case, such as the logic described by the basic course of action or a portion of the basic course of action, plus one or more alternate scenarios. The logic of a usage scenario may also be a pass through the logic contained in several use cases.

2. **The logic of methods:** Sequence diagrams can be used to explore the logic of a complex operation, function, or procedure. One way to think of sequence diagrams, particularly highly detailed diagrams, is as visual object code.

3. **The logic of services:** A service is effectively a high-level method, often one that can be invoked by a wide variety of clients. This includes web-services as well as business transactions implemented by a variety of technologies.

4.4 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as in operation of the system. The control flow is drawn from one operation to another.

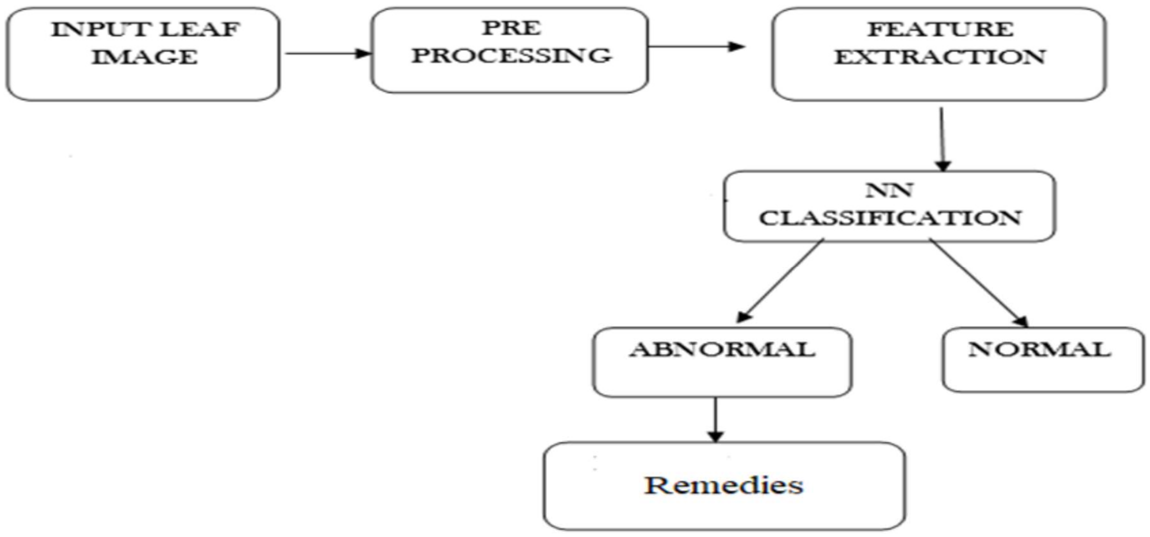


Figure 6: Activity Diagram

The above flow represents the flow from one activity to another activity, the activity starts from input leaf image through digital camera, and then input leaf is pre-processed and extract the features like color, shape, texture and so on. Now, the processed image is classified as Normal or Abnormal, if Abnormal is found in the leaf, then remedies will be suggested.

CHAPTER 5 : PROPOSED METHODOLOGY

PROPOSED METHODOLOGY

This part of the report illustrates the approach employed to classify the leaves into diseased or healthy and if the leaf is diseased, name of the disease is mentioned along with the remedies. Our methodology primarily revolves around the following five steps.

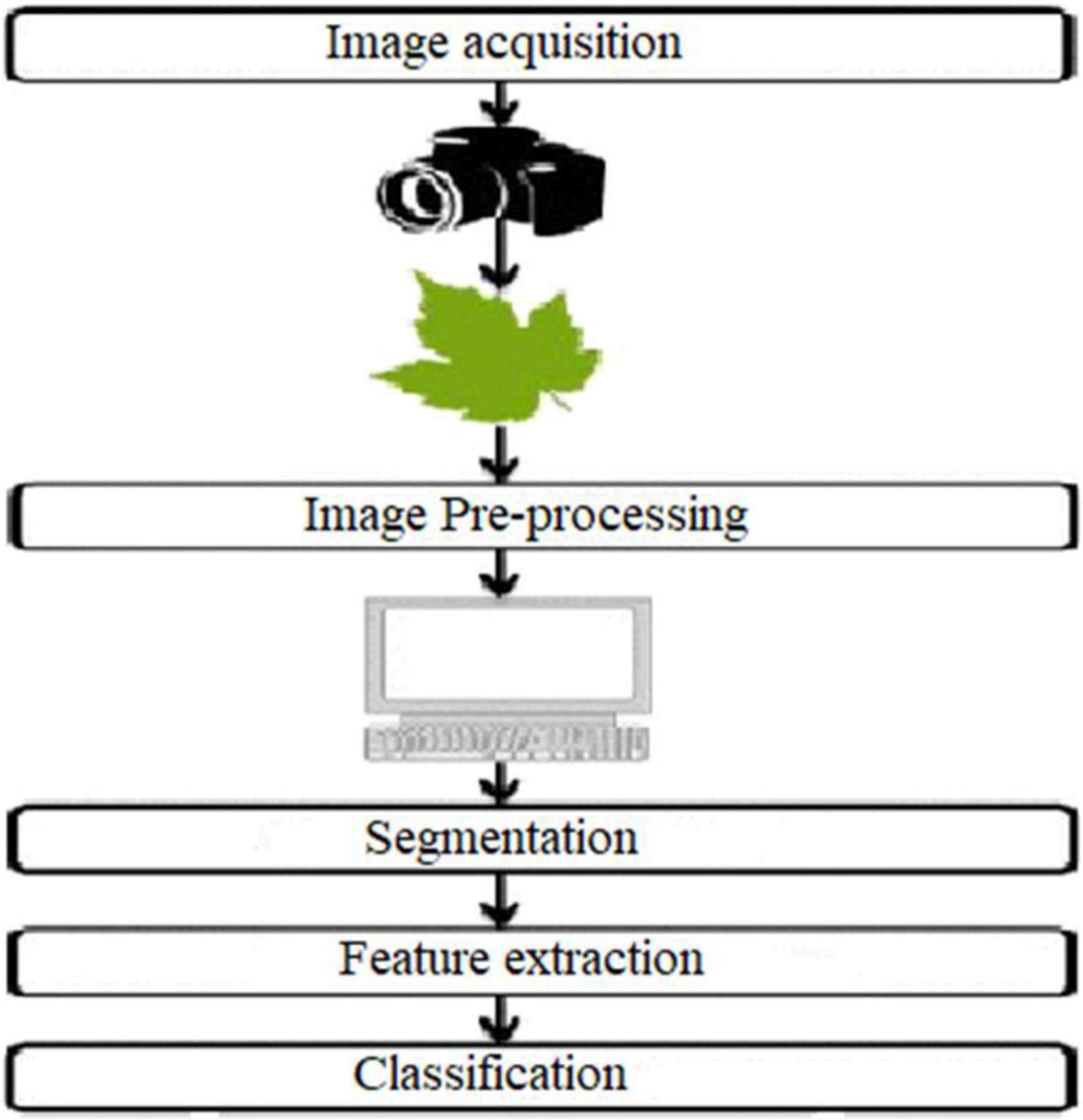


Figure 7: System Architecture

Algorithm written below illustrated the step by step approach for the proposed image recognition and segmentation processes:

1) Image acquisition is the very first step that requires capturing an image with the help of a digital camera.

2) Pre-processing of input image to improve the quality of image and to remove the undesired distortion from the image. Clipping of the leaf image is performed to get the interested image region and then image smoothing is done using the smoothing filter. To increase the contrast Image enhancement is also done.

3) Mostly green colored pixels, in this step, are masked. In this, we computed a threshold value that is used for these pixels. Then in the following way mostly green pixels are masked: if pixel intensity of the green component is less than the pre-computed threshold value, then zero value is assigned to the red, green and blue components of the this pixel.

4) In the infected clusters, inside the boundaries, remove the masked cells.

5) Obtain the useful segments to classify the leaf diseases.

5.1 Image Acquisition

The initial process is to collect the data from the public repository. It takes the image as input for further processing. We have taken most popular image domains so that we can take any formats like .bmp, .jpg, .gif as input to our process.

The image is captured, scanned and converted into a manageable entity. This process is known as image acquisition. During a test-phase, we acquire a series of color images using a digital scanner so as to acquire a single image of leaf. The color images were digitized to produce RGB digital color images.

5.2 Image Pre-processing

As the images are acquired from the real field it may contain dust, spores and water spots as noise. The purpose of data pre-processing is to eliminate the noise in the image, so as to adjust the pixel values. It enhances the quality of the image.

The main aim of Pre-processing is to suppress unwanted image data and to enhance some important image features. It includes RGB to Gray conversion, image resizing and median filtering. Here color image is converted to gray scale image to make the image device independent. The image is then resized to a size of 256*256. Then median filtering is performed on the image to remove the noise. The digital version of the rotten leaf sample consists of about 30% of leaf area and rest 70% is the background. Thus the redundant background requires high disk storage space and utilizes CPU time in the segmentation process. In order to have efficient disk storage and achieve fast processing speed the digital image of the leaf sample is cropped into a smaller dimension of size 16x20sq, cm. Thus the pre-processing step introduced saves about 30% of disk storage space and increases the CPU processing 1.4 times. The cropping process does not introduce any loss in the region of interest, i.e. the selected leaf sample. After pre-processing stage, the digital version of the sample leaf image consists about 70% of leaf area part and rest 30% as background. For getting better results in further steps, image pre-processing is required because dust, dewdrops, insect's excrements may be present on the plant; these things are considered as image noise. Furthermore, captured images may have distortion of some water drops and shadow effect, which could create problems in the segmentation and feature extraction stages. Effect of such distortion can be weakened or removed using different noise removal filters. There may be low contrast in captured images; for such images contrast enhancement algorithms can be used. Sometimes background removal techniques may also be needed in case of region of interest needs to be extracted. In case of noise such as salt and pepper, a median filter can be used. Weiner filter can be used to remove a blurring effect. In case of the images captured using high definition cameras, the size of the pictures might be very large, for that reduction of image size is required. Also, image reduction helps in reducing the computing memory power.

5.3 Segmentation

Image segmentation is the third step in our proposed method. The segmented images are clustered into different sectors using Otsu classifier and k-mean clustering algorithm. Before clustering the images, the RGB color model is transformed into Lab color model. The advent of Lab color model is to easily cluster the segmented images.

5.4 Feature extraction

In the proposed approach, the method adopted for extracting the feature set is called the Color Co-occurrence Method or CCM method in short. It is a method, in which both the color and texture of an image are taken into account, to arrive at unique features, which represent that image.

5.5 Classification

Convolutional neural networks are used in the automatic detection of leaves diseases. CNN is chosen as a classification tool due to its well-known technique as a successful classifier for many real applications.

CNN architectures vary with the type of the problem at hand. The proposed model consists of three convolutional layers each followed by a max-pooling layer. The final layer is fully connected MLP. ReLu activation function is applied to the output of every convolutional layer and fully connected layer.

The first convolutional layer filters the input image with 32 kernels of size 3x3. After max-pooling is applied, the output is given as an input for the second convolutional layer with 64 kernels of size 4x4. The last convolutional layer has 128 kernels of size 1x1 followed by a fully connected layer of 512 neurons. The output of this layer is given to Softmax function which produces a probability distribution of the four output classes.

Gray Level Co-occurrence Matrix is created from gray scale images and used to describe the shape feature. The Gray Level Co-occurrence Matrix is based on the repeated occurrence of gray-level configuration in the texture. The spatial gray dependence matrix is used for texture analysis. A spatial gray dependence matrix is created based on hue, saturation and intensity. Run Length Matrix (RLM) is another type of matrix. Same gray pixel values are the part of run and those gray values forms a two dimensional matrix.

Example, RM- Q(x, y) x represents gray values and y represents run length.

5.6 Convolutional Neural Network (ConvNet) Algorithm

1) A convolutional neural network is a class of deep neural network, most commonly applied to analysing visual imagery.
2) A Convolutional Neural Network is a Machine Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

INPUT LAYER

- In the figure below, we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

- The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

- This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets

.

CONVOLUTION LAYER -- The Kernel

- In the below demonstration, the green section resembles our 5x5x1 input image. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix{{1,0,1},{0,1,0},{1,0,1}}.
- The Kernel shifts 9 times because of Stride Length = 1, every time performing a matrix multiplication operation between K and the portion P of the image.

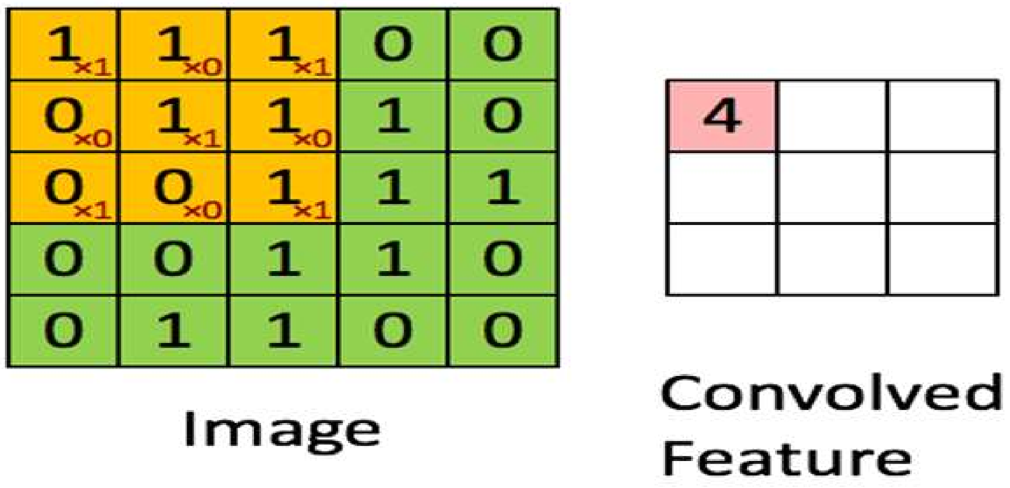


Figure 8:convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

- The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

POOLING LAYER

- The Pooling layer is responsible for reducing the spatial size of the Convolved Feature.
- It is even useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

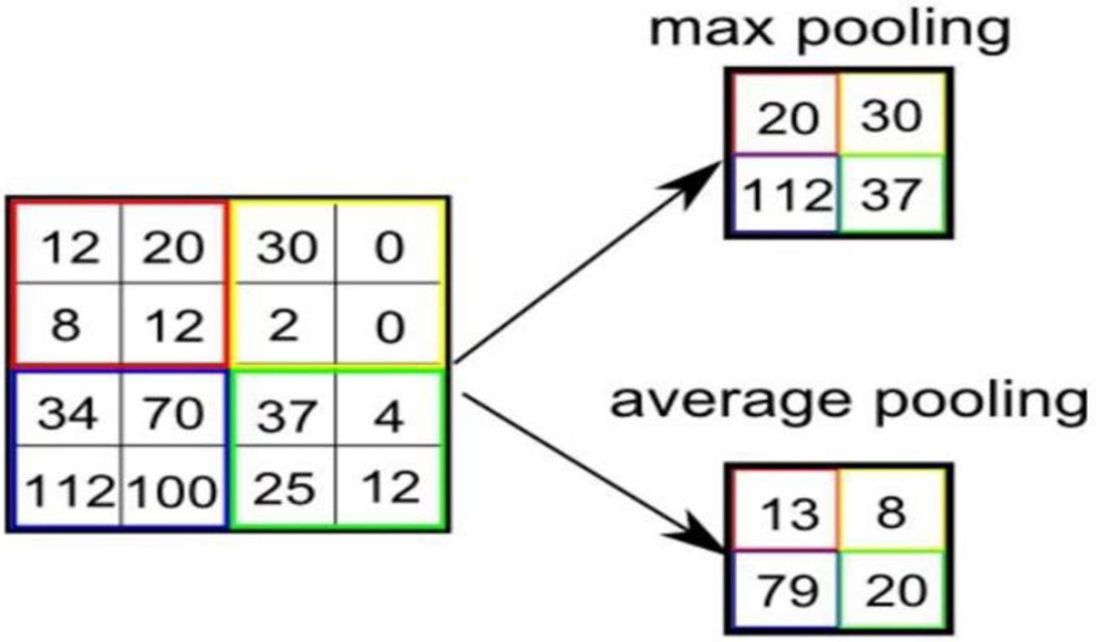


Figure 9: selecting maximum and average pooling matrix

- There are two types of Pooling: Max Pooling and Average Pooling.
- Max Pooling returns the maximum value from the portion of the image covered by the Kernel.
- On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

CHAPTER 6: MODELING

MODELING

6.1 Model Architecture

We’ll build our model from scratch. After experimenting on different models, I found the below architecture performing better.

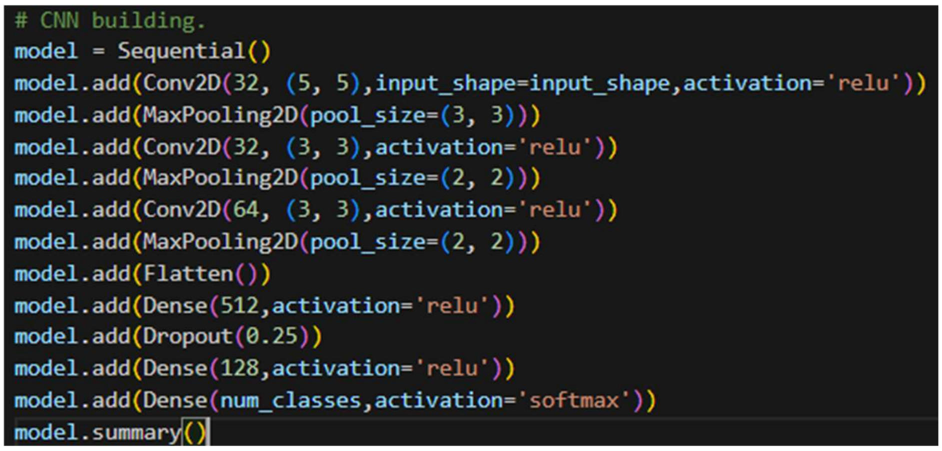
Note that we didn’t scaled our images when loading, because we added a Rescaling layer in our model to do it. This layer automatically scales unseen images, which is helpful for deployment.

We have a Softmax layer at the end to output the probabilities of classes.

Using CNN, the sequential model is formed. For this, we have used rectified linear units. SoftMax is also used like an activation for forecasting

$$P(x) = \frac{e^{x^T w^l}}{\sum_{k=1}^{k} e^{x^T w^l}}$$

which depends on the maximum likelihoods. The equation for the SoftMax function is given below:
Here, XT W signifies X and W's internal product.



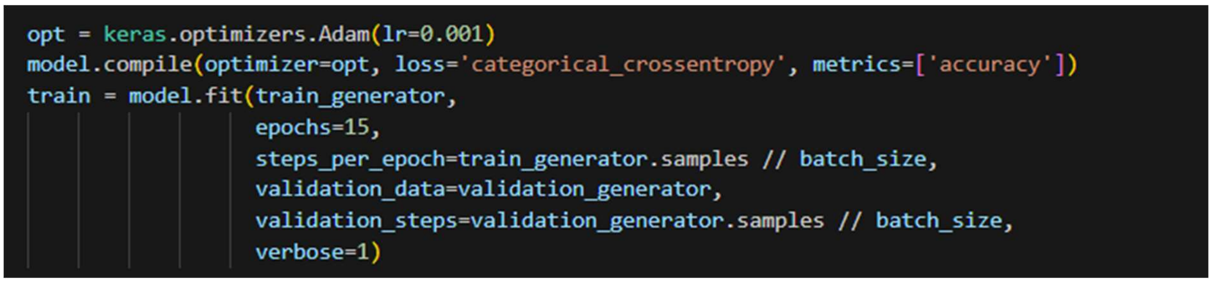
```
# CNN building.
model = Sequential()
model.add(Conv2D(32, (5, 5),input_shape=input_shape,activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(32, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128,activation='relu'))
model.add(Dense(num_classes,activation='softmax'))
model.summary()
```

Figure 10: CNN Model Architecture

6.2 Model training

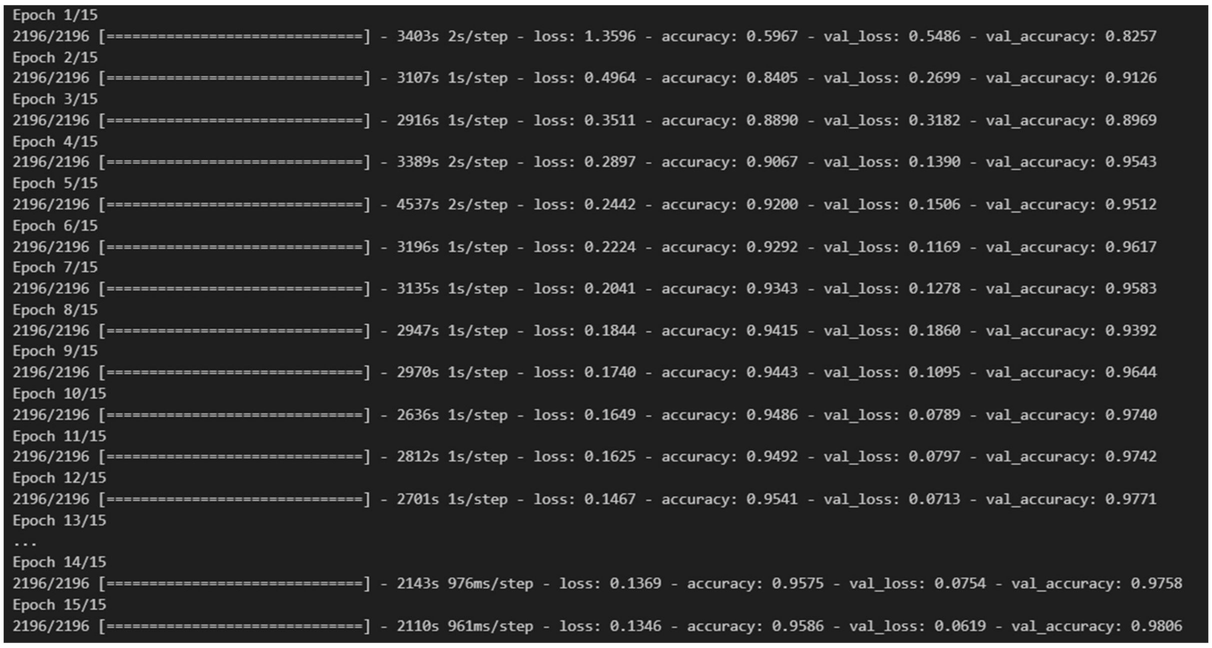
We have everything to train our model, We use Adam algorithm for optimization and Early Stopping call back to avoid over fitting.

We'll train our model for 10 epochs and store the training history in the history variable.



```
opt = keras.optimizers.Adam(lr=0.001)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
train = model.fit(train_generator,
                  epochs=15,
                  steps_per_epoch=train_generator.samples // batch_size,
                  validation_data=validation_generator,
                  validation_steps=validation_generator.samples // batch_size,
                  verbose=1)
```

Figure 11:Model Training



```
Epoch 1/15
2196/2196 [==============================] - 3403s 2s/step - loss: 1.3596 - accuracy: 0.5967 - val_loss: 0.5486 - val_accuracy: 0.8257
Epoch 2/15
2196/2196 [==============================] - 3107s 1s/step - loss: 0.4964 - accuracy: 0.8405 - val_loss: 0.2699 - val_accuracy: 0.9126
Epoch 3/15
2196/2196 [==============================] - 2916s 1s/step - loss: 0.3511 - accuracy: 0.8890 - val_loss: 0.3182 - val_accuracy: 0.8969
Epoch 4/15
2196/2196 [==============================] - 3389s 2s/step - loss: 0.2897 - accuracy: 0.9067 - val_loss: 0.1390 - val_accuracy: 0.9543
Epoch 5/15
2196/2196 [==============================] - 4537s 2s/step - loss: 0.2442 - accuracy: 0.9200 - val_loss: 0.1506 - val_accuracy: 0.9512
Epoch 6/15
2196/2196 [==============================] - 3196s 1s/step - loss: 0.2224 - accuracy: 0.9292 - val_loss: 0.1169 - val_accuracy: 0.9617
Epoch 7/15
2196/2196 [==============================] - 3135s 1s/step - loss: 0.2041 - accuracy: 0.9343 - val_loss: 0.1278 - val_accuracy: 0.9583
Epoch 8/15
2196/2196 [==============================] - 2947s 1s/step - loss: 0.1844 - accuracy: 0.9415 - val_loss: 0.1860 - val_accuracy: 0.9392
Epoch 9/15
2196/2196 [==============================] - 2970s 1s/step - loss: 0.1740 - accuracy: 0.9443 - val_loss: 0.1095 - val_accuracy: 0.9644
Epoch 10/15
2196/2196 [==============================] - 2636s 1s/step - loss: 0.1649 - accuracy: 0.9486 - val_loss: 0.0789 - val_accuracy: 0.9740
Epoch 11/15
2196/2196 [==============================] - 2812s 1s/step - loss: 0.1625 - accuracy: 0.9492 - val_loss: 0.0797 - val_accuracy: 0.9742
Epoch 12/15
2196/2196 [==============================] - 2701s 1s/step - loss: 0.1467 - accuracy: 0.9541 - val_loss: 0.0713 - val_accuracy: 0.9771
Epoch 13/15
...
Epoch 14/15
2196/2196 [==============================] - 2143s 976ms/step - loss: 0.1369 - accuracy: 0.9575 - val_loss: 0.0754 - val_accuracy: 0.9758
Epoch 15/15
2196/2196 [==============================] - 2110s 961ms/step - loss: 0.1346 - accuracy: 0.9586 - val_loss: 0.0619 - val_accuracy: 0.9806
```

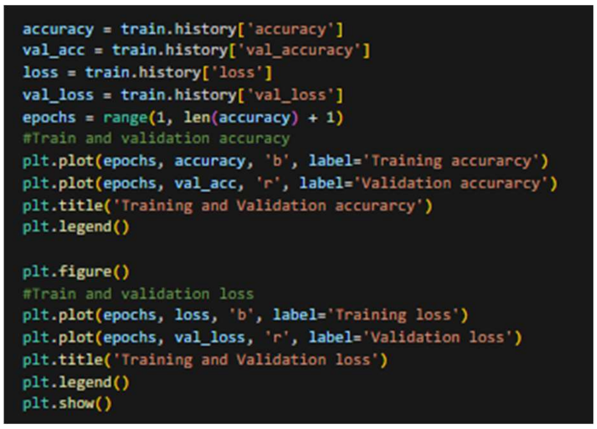
Figure 12: Accuracy

CHAPTER 7: MODEL EVALUATION

MODEL EVALUATION

7.1 Training history

Let’s look at Loss and Accuracy of the model at each epoch of training.



```
accuracy = train.history['accuracy']
val_acc = train.history['val_accuracy']
loss = train.history['loss']
val_loss = train.history['val_loss']
epochs = range(1, len(accuracy) + 1)
#Train and validation accuracy
plt.plot(epochs, accuracy, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```

Figure 13: Graph plot

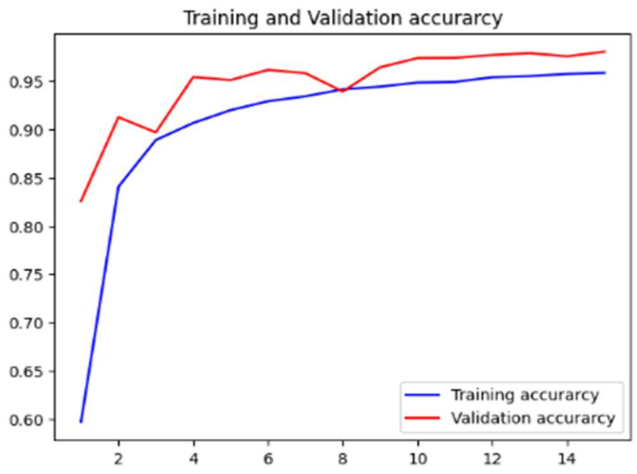


Figure 14: Training vs Validation Accuracy Graph

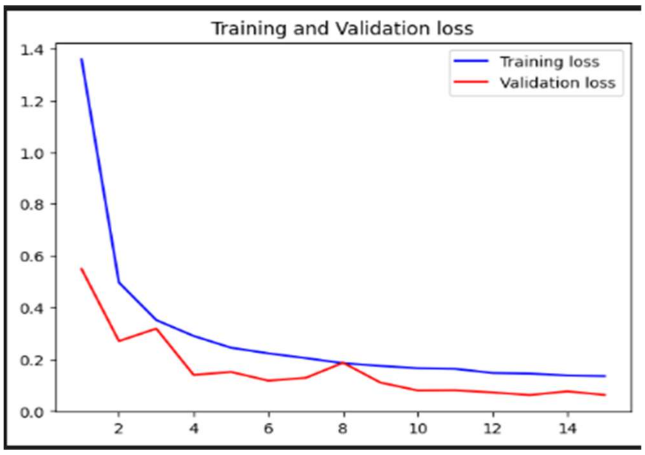


Figure 15: Training and Validation Loss Graph

7.2 Confusion matrix

By visualizing the confusion matrix, an individual could determine the accuracy of the model by observing the diagonal values for measuring the number of accurate classification.

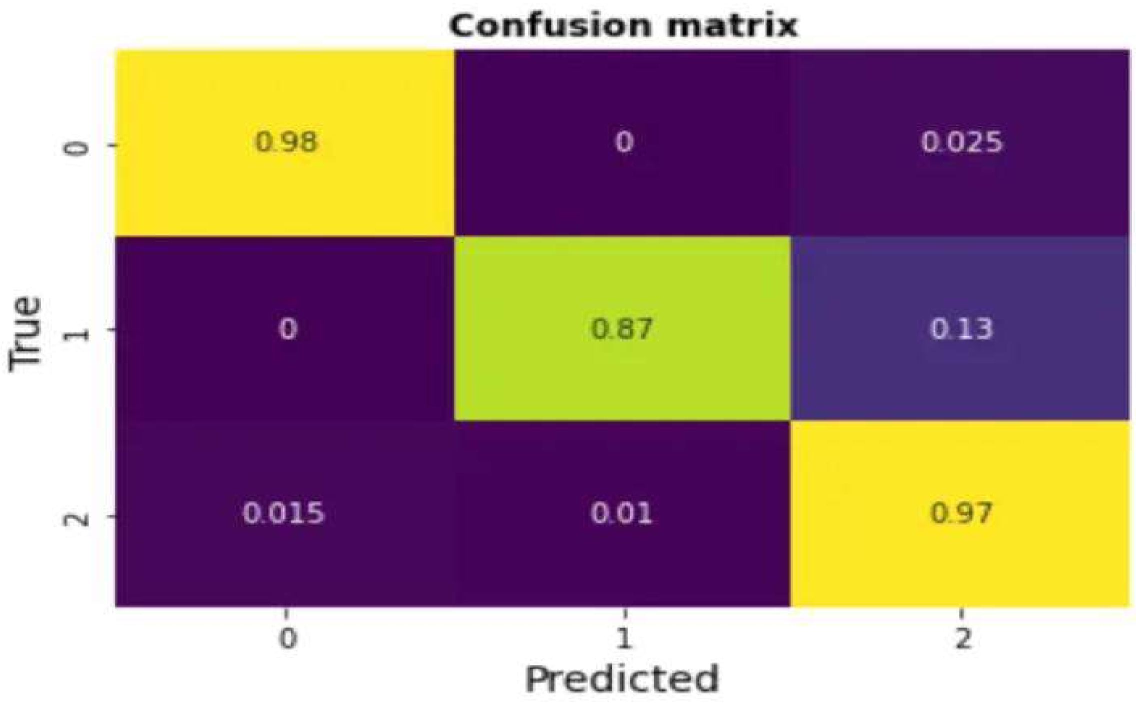
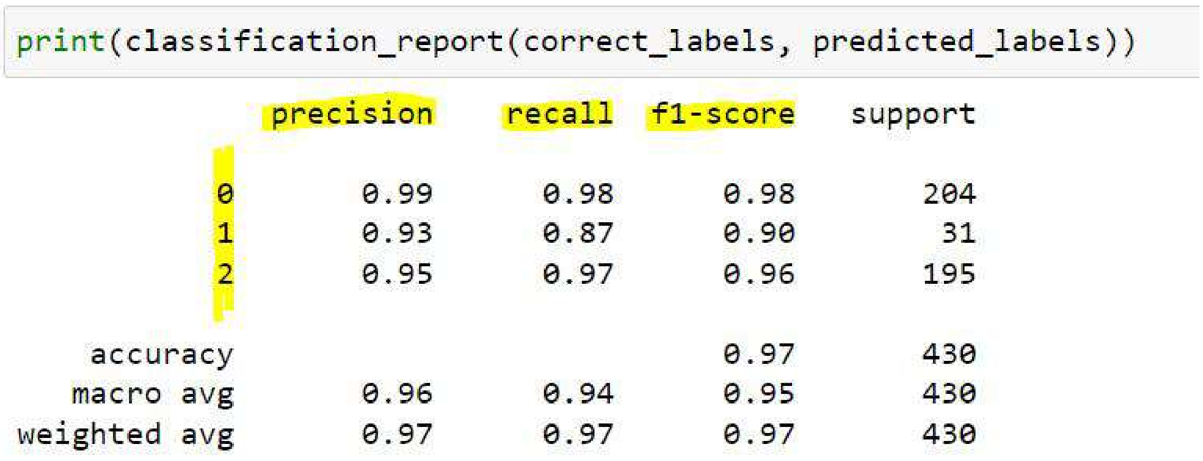


Figure 16: Confusion Matrix

7.3 Classification Report

Let’s look at some other metrics like precision, recall and f1-score.



```
print(classification_report(correct_labels, predicted_labels))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       204
           1       0.93      0.87      0.90        31
           2       0.95      0.97      0.96       195

    accuracy                           0.97       430
   macro avg       0.96      0.94      0.95       430
weighted avg       0.97      0.97      0.97       430
```

Figure 17: Classification Report

CHAPTER 8: APPLICATION DEVELOPMENT

APPLICATION DEVELOPMENT

8.1 Import Statements

- In the code, various packages are imported, including **dart:io**,
- **package:flutter/material.dart,**
- **package:image_picker/image_picker.dart,**
- **package:tflite_v2/tflite_v2.dart,** and
- custom Dart files like **community.dart, weather_display.dart,** and **Profile.dart**. These packages are used for various functionalities and widgets.

8.2 WeatherData Class

- A custom class WeatherData is defined to represent weather-related data with attributes like temperature and weatherCondition.

8.3 MyApp Class

- The MyApp class is a StatelessWidget that represents the main application. It sets the theme and the home page of the app.

8.4 ImagePickerDemo Class

- This is the main stateful widget of the application. It represents the app's UI and functionality.
- It includes instance variables for managing the image, recognition results, and the current page index.
- A WeatherData object is also defined, though it seems to be a placeholder for weather data.

8.5 initState Method

- Inside the initState method, the model for image recognition is loaded using the Tflite.loadModel method. This is done when the widget is first created.

8.6 _pickImage Method

- _pickImage is a method that allows the user to select an image either from the gallery or by taking a picture with the camera. The selected image is stored in the _image variable, and image recognition is triggered by calling detectImage.

8.7 detectImage Method

- detectImage takes a File object representing the selected image and runs image recognition on it using the loaded TFLite model. The recognition results are stored in _recognitions and displayed as text in the UI.

8.8 _onTabTapped Method

- _onTabTapped is called when the user taps on one of the bottom navigation bar items, changing the current page index.

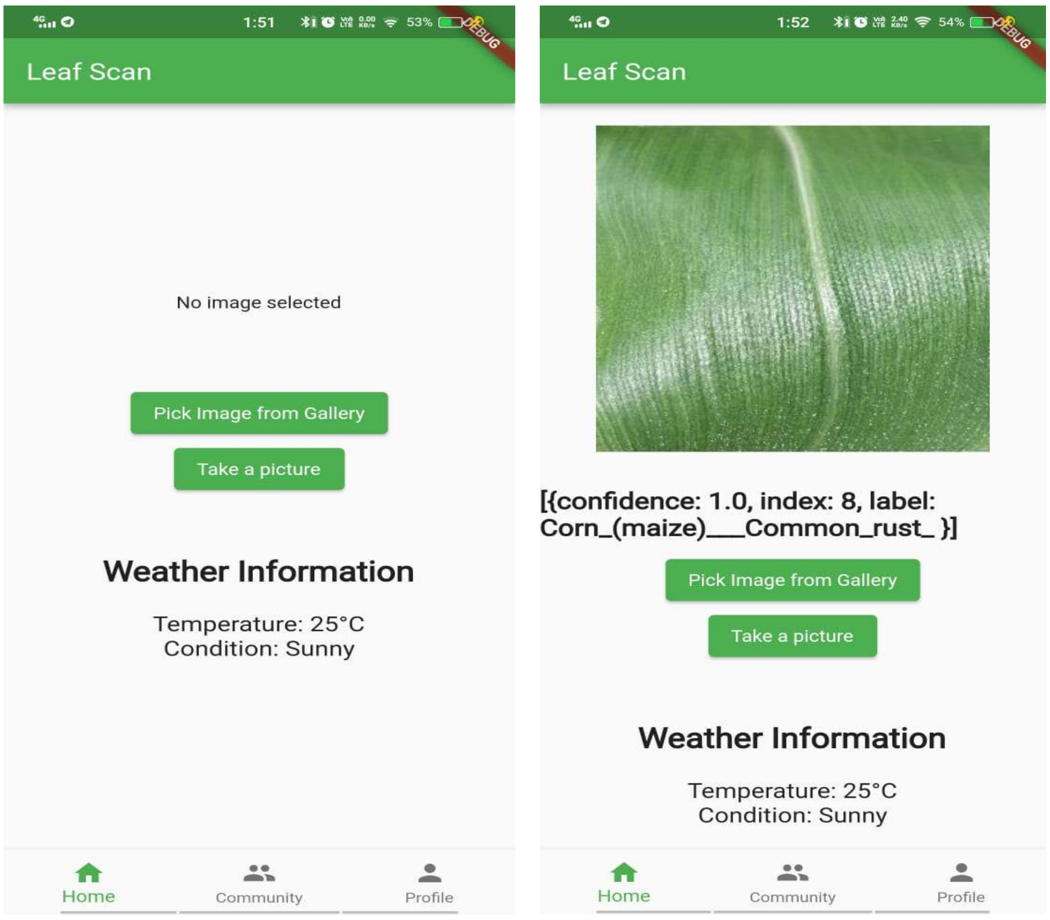
8.9 build Method

- The build method defines the app's UI using a Scaffold widget.
- There are three pages represented by a list of widgets (pages). The current page is determined by _currentIndex.
- The selected page is displayed in the body of the Scaffold.
- The bottom navigation bar allows the user to switch between the home, community, and profile pages.

8.10 Recognitions Display

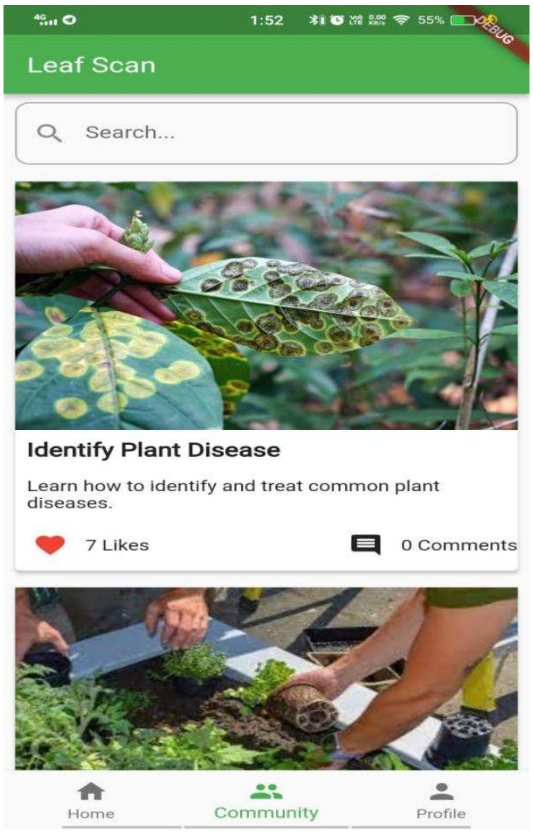
- Recognitions are displayed below the selected image in a Text widget. The text is styled to be bold and have a larger font size for better visibility.

8.11 Application images



```
[{confidence: 1.0, index: 8, label:
Corn_(maize)___Common_rust_}]
```

Figure 18: Home Page

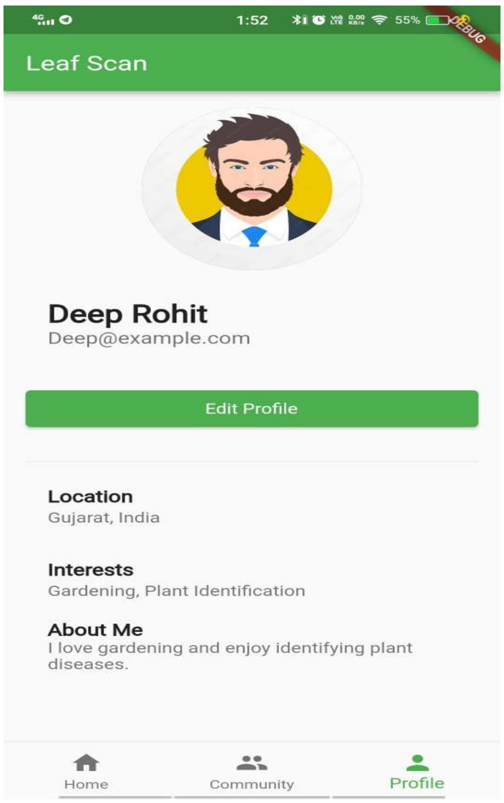


Identify Plant Disease

Learn how to identify and treat common plant diseases.

7 Likes 0 Comments

Figure 19: Community Page



Deep Rohit
Deep@example.com

Location
Gujarat, India

Interests
Gardening, Plant Identification

About Me
I love gardening and enjoy identifying plant diseases.

Figure 20: Profile Page

CHAPTER 9: SYSTEM TESTING

SYSTEM TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. There are various types of testing, explained in the following sections.

9.1 Testing Objective

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test one that uncovers an as –yet undiscovered error.

The above objectives imply a dramatic change in viewpoint. They move counter to the commonly held view that a successful test is one in which no errors are found. Testing cannot show the absence of detects, it can only show that software errors are present.

9.2 White Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, or structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality as in Black Box Testing Using white box testing methods, the software engineer can derive test cases that:

- Guarantee that all independent paths with in a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their validity.
- Logical errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed.
- Often believe that logical path not likely to execute when, in fact, it may be executed on regular basis.

9.3 Levels of Testing

The different levels of testing that are to be conducted are:

- Code Testing
- Program Testing
- System Testing

- ➢ **Code Testing:** The code test has been conducted to test the logic of the program. Here, we have tested with all possible combinations of data to find out logical errors. The code testing is done thoroughly with all possible data available with library.

- ➢ **Program Testing:** Program testing is also called unit testing. The modules in the system are integrated to perform the specific function. The modules have been tested independently, later Assembled and tested thoroughly for integration between different modules.

- **System Testing:** System testing has been conducted to test the integration of each module in the system. It is used to find discrepancies between the system and its original objective. It is found that there is an agreement between current specifications and system documentation. Software Testing is carried out in three steps.

  The first step includes unit testing where in each module is tested to provide his correctness, validity and also determine any missing operations. Errors are noted down and corrected immediately. Unit testing is the import and major part of the project. So errors are rectified easily in particular module and program clarity is increased. In this project entire system is divided into several modules and is developed individually. So unit testing is conducted to individual modules.

  The second step includes integration testing. It need not be the case, the software whose modules when run individually and showing perfect results, will also show perfect results when run as a whole. The individual modules are clipped under this major module and tested again and the results are verified.

  The final step involves validation and testing which determines the software functions as the user expected. Here also there may be some modifications. In the completion of the project it is satisfied fully by the user.

9.4 Integration Test Plan

Data can be lost across an interface, one module can have an adverse effort on the other sub functions, when combined, may not produce the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. The testing was done with sample data. The developed system has run successfully for this sample data. The need for integrated test is to find the overall system performance.

9.5 Unit Testing

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example, the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Function Name	**Tests Results**
Uploading Image	Tested for uploading different types and sizes of images.
Detecting Disease	Tested for different images of healthy and diseased leaves.

Table 1: Unit testing Table

9.6 Output Testing

After performance of the validation testing, the next step is output testing. The output displayed or generated by the system under consideration is tested by asking the user about the format required by system.

Functionality to be tested	Input	**Tests Results**
Working of Choose File option	User convenience to access images stored	Different folders where images are stored can be uploaded
Working of View Remedies	User need to click the Remedies button	Details of disease remedies are displayed.

Table 2: Output Testing Table

CHAPTER 10: RESULTS AND DISCUSSION

RESULTS AND OUTCOMES

In our project, we developed a versatile multi-stage deep learning model designed to detect various crop diseases, spanning a total of 14 different classes. Our main goal is to improve crop productivity and quality by effectively controlling biological factors that lead to significant yield losses. First, we extract features from the leaf images of these 14 different crop classes. Then, at a second-order level, we use specialized convolutional neural networks (CNNs) to accurately identify and classify diseases specific to each plant category. Furthermore, our model considers the influence of environmental variables on plant leaf diseases, increasing its adaptability and accuracy in different agricultural situations. For validation, we performed an extensive evaluation using separate datasets across 14 different cultural classes. The results clearly demonstrate the superior performance of our method compared to alternative techniques and demonstrate its versatility and effectiveness on a wide range of crops. Our approach achieved an excellent accuracy of 98.06% while exhibiting high precision, recall, F1 score, and Roc curve for these diverse crops. This highlights the robustness and wide applicability of our model in effectively detecting diseases in different agricultural environments.

One of the most notable advantages of our approach is its adaptability. Our model is capable of handling a wide range of crops, has a lightweight architecture with minimal parameters, and differs from current state-of-the-art methods. This not only reduces computational costs but also significantly increases processing speed, making it a highly practical and resource-efficient solution for detecting crop foliar diseases in a wide range of agricultural environments.

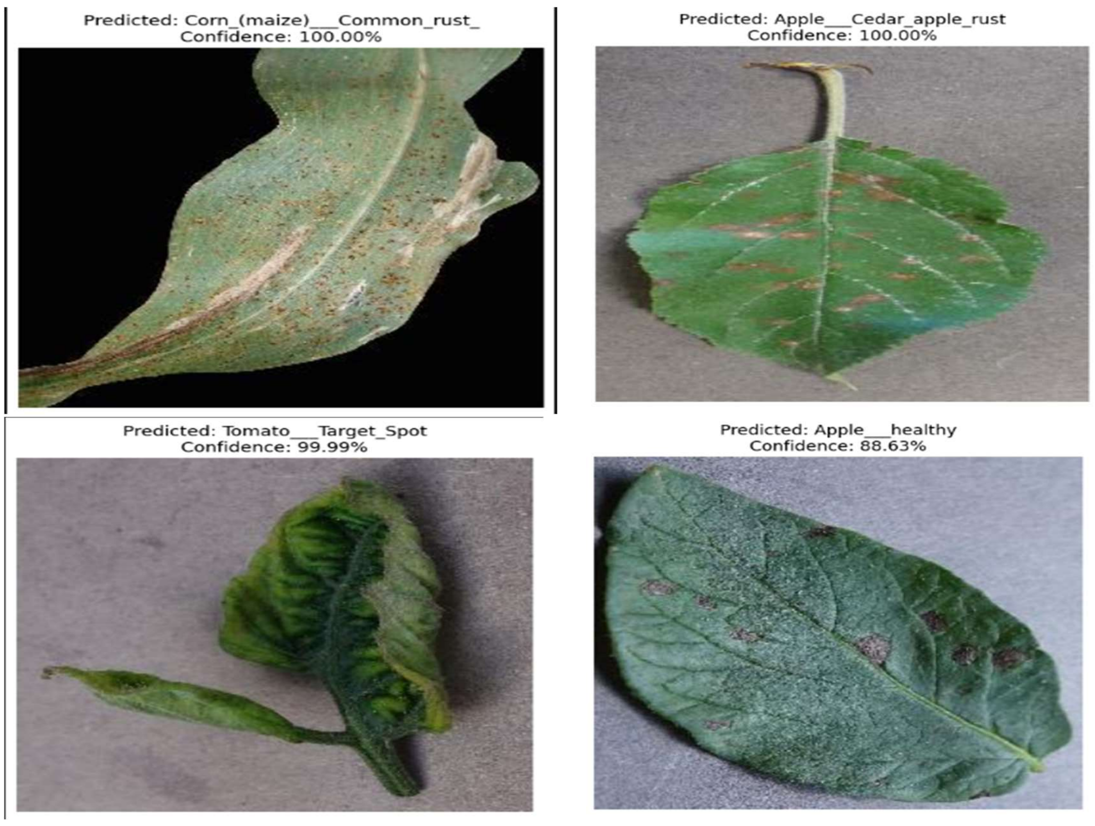

Predicted: Corn_(maize)___Common_rust_
Confidence: 100.00%
Predicted: Apple___Cedar_apple_rust
Confidence: 100.00%
Predicted: Tomato___Target_Spot
Confidence: 99.99%
Predicted: Apple___healthy
Confidence: 88.63%

Figure 21: Results

CONCLUSION AND FUTURE WORK

CONCLUSION

In conclusion, this study presents a pioneering approach in the field of plant pathology by successfully devising disease classification techniques for the detection of various plant leaf diseases. Through the development of a robust deep learning model, we have demonstrated its efficacy in automatically detecting and classifying diseases across 14 different plant species, encompassing a total of 38 distinct plant classes. With an impressive global model validation accuracy of 97%, this CNN-based system holds immense promise for practical implementation in real-world agricultural settings. The potential impact of this research on enhancing disease classification and enabling early detection cannot be overstated, ultimately leading to substantial improvements in crop yields. This paper stands as a significant contribution to the field of plant health management, providing valuable insights that will undoubtedly shape the future of agricultural practices.

FUTURE WORK

- To improve recognition rate of final classification process hybrid algorithms like Artificial Neural Network, Bayes classifier, Fuzzy Logic can also be used.
- An extension of this work will focus on automatically estimating the severity of the detected disease.
- As future enhancement of the project is to develop the open multimedia (Audio/Video) about the diseases and their solution automatically once the disease is detected.

Reference

1. Bharath, S., Kumar, K.V., Pavithran, R., Malathi, T.: Detection of plant leaf diseases using CNN. In: Computer Science and Engineering. , Chennai. India (2020).
2. Zeng, W., Li, M.: Crop leaf disease recognition based on Self-Attention convolutional neural network. Computers and Electronics in Agriculture. 172, (2020).
3. Paymode, A.S., Malode, V.B.: Transfer learning for multi-crop leaf disease image classification using convolutional neural network VGG. Artificial Intelligence in Agriculture. 6, 23–33 (2022). https://doi.org/10.1016/j.aiia.2021.12.002.
4. Pantazi, X.E., Moshou, D., Tamouridou, A.A.: Automated leaf disease detection in different crop species through image features analysis and One Class Classifiers. Comput. Electron. Agric. 156, 96–104 (2019). https://doi.org/10.1016/j.compag.2018.11.005.
5. Jaware, T.H., Badgujar, R.D., Patil, P.G.: Crop disease detection using image segmentation. World Journal of Science and Technology. 2, 190–194 (2012).
6. Sardogan, M., Tuncer, A., Ozen, Y.: Plant leaf disease detection and classification based on CNN with LVQ algorithm. In: 2018 3rd International Conference on Computer Science and Engineering (UBMK). IEEE (2018).
7. Memon, Muhammad Suleman, Pardeep Kumar, and Rizwan Iqbal. "Meta deep learn leaf disease identification model for cotton crop." Computers 11, no. 7 (2022): 102.
8. Liu, J., Wang, X.: Plant diseases and pests detection based on deep learning: a review. Plant Methods. 17, 22 (2021). https://doi.org/10.1186/s13007-021-00722-9.
9. Mohanty, S.P., Hughes, D.P., Salathé, M.: Using deep learning for image-based plant dis-ease detection. Front. Plant Sci. 7, 1419 (2016). https://doi.org/10.3389/fpls.2016.01419.
10. Kulkarni, P., Karwande, A., Kolhe, T., Kamble, S., Joshi, A., Wyawahare, M.: Plant disease detection using image processing and machine learning, http://arxiv.org/abs/2106.10698, (2021).
11. Eunice, Jennifer, Daniela Elena Popescu, M. Kalpana Chowdary, and Jude Hemanth. "Deep learning-based leaf disease detection in crops using images for agricultural applications." Agronomy 12, no. 10 (2022): 2395.
12. Gavhale, K.R., Gawande, U.: An overview of the research on plant leaves disease detection using image processing techniques. Iosr Journal of Computer Engineering (iosrjce). 16, 10–16 (2014).
13. Pantazi, X.E., Moshou, D., Tamouridou, A.A.: Automated leaf disease detection in different crop species through image features analysis and Class Classifiers. Computers and electron-ics in agriculture. 156, 96–104 (2019).
14. https://www.youtube.com/watch?v=148eu_foNo8
15. https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset
16. https://www.researchgate.net/publication/314436486_An_Overview_of_the_Research_on_Plant_Leaves_Disease_detection_using_Image_Processing_Techniques